JOURNAL OF COMPUTER AND SYSTEM SCIENCES 28, 345–358 (1984)

On the Complexity of Iterated Shuffle*

MANFRED K. WARMUTH† AND DAVID HAUSSLER‡

Department of Computer Science,
University of Colorado, Boulder, Colorado 80309

Received April 7, 1981; revised October 27, 1982

It is demonstrated that the following problems are *NP* complete:

(1) Given words w and $w_1, w_2, \ldots, w_n$, is w in the shuffle of $w_1, w_2, \ldots, w_n$?

(2) Given words w and v, is w in the iterated shuffle of v?

From these results we show that the languages $\{\$w\¢w^R : w \in \{a, b\}^*\}^{\circledast}$, $\bigcup_{w\in\{a,b\}^*} (\$w)^{\circledast}$, $\{ab^n cde^n f : n \geqslant 0\}^{\circledast}$, and $\{a^{n+1}b^n c^n f^n : n \geqslant 0\}^{\circledast}$ are *NP* complete, where $^{\circledast}$ denotes the iterated shuffle. By representing these languages in various ways using the operations shuffle, iterated shuffle, union, concatenation, intersection, intersection with a regular set, non-erasing homomorphism and inverse homomorphism, results on the complexity of language classes generated using various subsets of these operations are obtained. Finally, it is shown that the iterated shuffle of a regular set can be recognized in deterministic polynomial time.

I. INTRODUCTION

The operations of shuffle and iterated shuffle have been used by numerous researchers to describe sequential computation histories of concurrent programs [1, 8, 10–12]. Recently, there has been some investigation into the language generating power of the operations of shuffle and iterated shuffle, when used in combination with the more conventional operations of union, concatenation, Kleene star, intersection, intersection with a regular set, non-erasing homomorphism and inverse homomorphism [4–6, 9, 13]. Jantzen [5, 6] has obtained numerous representations of the recursively enumerable languages and of the homomorphic images of the computation sequence sets of petri nets as the closure of various elementary language classes under certain combinations of the above operations, but allowing arbitrary homomorphisms or other forms of erasing via cancellation. On the other

* This research was supported in part by NSF Grant MCS 79-03838, the University of Colorado doctoral fellowship program, the Fulbright Commission of West Germany, and grants from Univac Corporation, and Storage Technology Corporation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

† Present address: Department of Computer and Information Sciences, University of California, Santa Cruz, California 95064.

‡ Present address: Department of Mathematics and Computer Science, University of Denver, Denver, Colorado 80208.

0022-0000/84 $3.00

hand, it is clear that the class *NP* is closed under the fundamental set of operations listed above, in which we have omitted the non-erasing homomorphism and all forms of cancellation. Thus the language classes generated from the finite languages using these length preserving operations will be subsets of *NP*. Jantzen's results using nonerasing homomorphisms leave it unclear at which point, if any, the languages generated using these operations become intractable.

In this paper we will be concerned with distinguishing those language classes generated by various combinations of the above operations which are recognized in deterministic polynomial time from those that contain *NP* complete languages. Intuitively, the operations of shuffle and iterated shuffle are natural condidates for producing the kind of "hiding of information" which generates *NP* complete languages. This intuition is borne out in what follows.

In Section II we give a small set of definitions. Then in Section III we present two basic *NP* complete problems. In the first, we are given a word w and words $w_1, w_2, \ldots, w_n$ and we ask whether w is in the shuffle of the words $w_1, w_2, \ldots, w_n$. This problem remains *NP* complete even if all words $w_1, w_2, \ldots, w_n$ are identical, which yields a second *NP* complete problem: for two words w and v, the problem of whether or not w is in the iterated shuffle of v is *NP* complete.

Using the above *NP* complete problems we show in Section IV that

$$\{\$w\¢w: w \in \{a, b\}^*\}^{\circledast}, \{\$w\¢w^R: w \in \{a, b\}^*\}^{\circledast}, \bigcup_{w \in \{a,b\}^*} (\$w)^{\circledast}, \{ab^n cde^n f: n \geqslant 0\}^{\circledast}$$

and $\{a^{n+1}b^n c^n d^n: n \geqslant 0\}^{\circledast}$ are *NP* complete, where $^{\circledast}$ denotes the iterated shuffle. Because of their simple structure, the last two languages can be expressed using various subsets of our fundamental set of operations. Hence we are able to use these languages to explore the complexity of the language classes generated by closing the finite sets under certain subsets of these basic operations. For instance, $\{ab^n cde^n f: n \geqslant 0\}^{\circledast} = (\{acdf, be\}^{\circledast} \cap ab^* cde^* f)^{\circledast}$, hence the closure of the finite languages under iterated shuffle and intersection with a regular set contains *NP* complete languages. A summary of the complexity results of this type is given in Table I.

We conclude Section IV by exploring the effect of alphabet size on the complexity of the problems and languages we have considered. Basically, we show that two letters are enough to produce the degree of complexity achieved using larger alphabets.

We note that Jantzen [6] has shown that $\{ab^n cde^n f: n \geqslant 0\}^{\circledast}$ is not a member of the subset of deterministic polynomial time languages accepted by real time one-way multi-counter machines. We show that this language is *NP* complete, thus we would expect that in fact $\{ab^n cde^n f: n \geqslant 0\}^{\circledast}$ is not in $\mathscr{P}$ at all.

This language also demonstrates that the iterated shuffle of a linear language can be *NP* complete. In Section V we give a deterministic polynomial time algorithm for the iterated shuffle of an arbitrary regular language. Using this result we show that the class *Shuf*, defined by Jantzen [5, 6] as the closure of the finite languages under the operations of union, shuffle, and iterated shuffle, is contained in $\mathscr{P}$. We also show

that the closure of the finite languages under any pair of operations consisting of iterated shuffle and one operation in the set {intersection, non-erasing homomorphism, inverse homomorphism, union, and Kleene star} is contained in $\mathscr{P}$. It remains open whether or not this is the case for the operations of iterated shuffle and concatenation.[1] We suspect that in fact the class *SE*, obtained by closing the class of finite sets under the operations union, concatenation, Kleene star, shuffle, and iterated shuffle, is contained in $\mathscr{P}$. This class has been examined in several papers including [6, 8, 12], usually being presented as the class of languages obtained by extending the regular expressions to include the operations of shuffle and iterated shuffle. The complexity of these "extended regular languages" remains a primary open problem in this area.

II. Definitions

Several different symbols for the operations shuffle and iterated shuffle have been used in the literature (see, e.g., [5, 10, 12]). We use the symbol $\odot$ for shuffle and $^{\circledast}$ for iterated shuffle, following Shaw [12], because they correspond nicely to the operations concatenation $\cdot$ and Kleene star *. Formally, we have the following definitions: Given a finite alphabet Σ and words $v, w \in \Sigma^*$, we define

$$v \odot w = \{v_1 w_1 v_2 w_2 \cdots v_k w_k : v_i, w_i \in \Sigma^* \text{ for } 1 \leqslant i \leqslant k, v = v_1 \cdots v_k \text{ and } w = w_1 \cdots w_k\},$$

$$w^{\circledast} = \{\lambda\}, \qquad w^{\textcircled{i+1}} = w^{\textcircled{i}} \odot w, \qquad \text{and} \qquad w^{\circledast} = \bigcup_{i=0}^{\infty} w^{\textcircled{i}}.$$

Given words $w_1, w_2, \ldots, w_k \in \Sigma^*$, we define $\prod_{i=1}^{k} w_i = w_1 w_2 \cdots w_k$ and $\textcircled{\Pi}_{i=1}^{k} w_i = w_1 \odot w_2 \odot \cdots \odot w_k$. We will extend the above operations from strings to languages in the usual manner.

All the polynomial reductions of our paper are reductions from the following *NP* complete problem:

3-Partition

Given: a sequence of natural numbers $S = \langle n_i : 1 \leqslant i \leqslant 3m \rangle$ such that $B = (\sum_{i=1}^{3m} n_i)/m \in \mathbb{N}$ and for each i, $1 \leqslant i \leqslant 3m$, $B/4 < n_i < B/2$.

Question: Can S be partitioned into m disjoint subsequences $S_1, \ldots, S_m$ such that for each k, $1 \leqslant k \leqslant m$, S_k has exactly three elements and $\sum_{n \in S_k} n = B$. This was the first problem proven to be *NP* complete in the strong sense [2, 3]. Thus it remains *NP* complete when the n_i are given in unary notation, which will be essential for our paper. The number B above will be referred to as the bound for S and the partition of S described above will be called a 3-partition.

[1] Jantzen [7] recently proved this class to be in $\mathscr{P}$.

III. Two *NP* Complete Problems

In this section we will demonstrate how the shuffle operation can be used to perform the partitioning and addition required to determine if a given instance of 3-partition has a solution. Our goal is to derive two fundamental *NP* complete problems involving the shuffle operation from 3-partition. First, we present a few properties of the shuffle operation which will be used extensively in this paper.

LEMMA 3.1. *Given a finite alphabet Σ containing the symbols a, b, and c, the words $v, x, w, w_1, w_2, \ldots, w_l \in \Sigma^*$, and natural numbers $n, n_1, n_2, \ldots, n_l$, for some $l > 0$, where x does not begin with a, b, or c and w_i does not begin with c, for i: $1 \leqslant i \leqslant l$, then*

(i) $a^k b^n c^k w \in x \odot \bigcirc\!\!\!\!\prod_{i=1}^{l} ab^{n_i}cw_i \Leftrightarrow$ *there exist distinct indices $j_1, \ldots, j_k$ such that $n = \sum_{r=1}^{k} n_{j_r}$ and $w \in x \odot (\bigcirc\!\!\!\!\prod_{r=1}^{k} w_{j_r}) \odot \bigcirc\!\!\!\!\prod_{i=1, i \notin \{j_1,\ldots,j_k\}}^{l} ab^{n_i}cw_i$.*

(ii) $a^n c^n w \in x \odot \bigcirc\!\!\!\!\prod_{i=1}^{l} a^{n_i}c^{n_i}w_i \Leftrightarrow$ *there exist distinct indicies $j_1, \ldots, j_k$ such that $n = \sum_{r=1}^{k} n_{j_r}$ and $w \in x \odot (\bigcirc\!\!\!\!\prod_{r=1}^{k} w_{j_r}) \odot \bigcirc\!\!\!\!\prod_{i=1, i \notin \{j_1,\ldots,j_k\}}^{l} a^{n_i}c^{n_i}w_i$.*

(iii) $a^{n+1}c^n w \in x \odot \bigcirc\!\!\!\!\prod_{i=1}^{l} a^{n_i+1}c^{n_i}w_i \Leftrightarrow$ *there exists an index k such that $n_k = n$ and $w \in x \odot \bigcirc\!\!\!\!\prod_{i=1, i \neq k}^{l} a^{n_i+1}c^{n_i}w_i$.*

Proof. (i) The words $w_1, w_2, \ldots, w_l$ do not start with the symbol c. Therefore, to account for the a's and c's, the prefix $a^k b^n c^k$ of $a^k b^n c^k w$ must be formed by shuffling exactly k prefixes of the $ab^{n_i}cw_i$'s of the form $ab^{n_i}c$ and the result follows.

(ii) Since the w_i's do not start with c, the prefix $a^n c^n$ of $a^n c^n w$ has to be created by shuffling prefixes of the $a^{n_i}c^{n_i}w_i$'s of the form $a^p c^q$, $p \geqslant q \geqslant 0$. Since n a's are needed followed by an equal amount of c's only prefixes of the form $a^p c^p$ can be used and the result follows.

(iii) To get the prefix $a^{n+1}c^n$ of $a^{n+1}c^n w$, prefixes of the $a^{n_i+1}c^{n_i}w_i$'s of the form $a^p c^q$, $p > q \geqslant 0$, have to be shuffled. Since there is only one more a than c in $a^{n+1}c^n$ at most one prefix can be used. ∎

LEMMA 3.2. *Let $S = \langle n_i : 1 \leqslant i \leqslant 3m \rangle$ be an instance of 3-partition with bound B. The following are equivalent:*

(i) *S has a 3-partition,*

(ii) $(a^3 b^B c^3)^m \in \bigcirc\!\!\!\!\prod_{i=1}^{3m} ab^{n_i}c$,

(iii) $(a^B b^B)^m \in \bigcirc\!\!\!\!\prod_{i=1}^{3m} a^{n_i}b^{n_i}$.

Proof. It is easily seen that (i) ⇒ (ii) and (i) ⇒ (iii). We simply merge using the shuffle operation each of the m triplets guaranteed by the partition to obtain the desired word. To see that (ii) ⇒ (i) we argue by induction on m. The case $m = 1$ is trivial. If $w = a^3 b^B c^3 (a^3 b^B c^3)^m \in \bigcirc\!\!\!\!\prod_{i=1}^{3(m+1)} ab^{n_i}c$ for some $S = \langle n_i : 1 \leqslant i \leqslant 3(m+1) \rangle$, an instance of 3-partition with bound B, then the initial $a^3 b^B c^3$ of w must be the result of shuffling exactly three words from $\langle ab^{n_i}c : 1 \leqslant i \leqslant 3(m+1) \rangle$. This follows directly from Lemma 3.1(i). Thus the remaining suffix $(a^3 b^B c^3)^m$ of w is formed by

shuffling the remaining $3m$ $ab^{n_i}c$'s. By hypothesis then, these remaining n_i's have a 3-partition. Hence the entire sequence S has a 3-partition and the inductive step is complete. The proof that (iii) $\Rightarrow$ (i) is similar. Assuming that $a^Bb^B(a^Bb^B)^m \in \textcircled{\Pi}_{i=1}^{3(m+1)} a^{n_i}b^{n_i}$ for S, B as above, again the initial a^Bb^B must be obtained from shuffling some words from $T = \langle a^{n_i}b^{n_i}: 1 \leqslant i \leqslant 3(m+1)\rangle$. Here we use Lemma 3.1(ii). The fact that in any instance of 3-partition we require that $B/4 < n_i < B/2$ for all i implies that we must shuffle exactly three words from T to obtain the initial a^Bb^B. Our result then follows by induction as above. ∎

As an immediate consequence of Lemma 3.2, we have

THEOREM 3.1. *Given a finite alphabet $\Sigma = \{a, b, c\}$ and words $w, w_1, \ldots, w_k \in \Sigma^*$, the problem of whether or not $w \in \textcircled{\Pi}_{i=1}^{k} w_i$ is NP complete.*

Proof. We use Lemma 3.2 to encode an instance of 3-partition into the above problem. Since 3-partition is strongly *NP* complete, we can assume the numbers are given in unary notation, thus the transformation will be polynomial. ∎

The above problem remains *NP* complete when we restrict ourselves to the case in which all of the w_i's are identical. To demonstrate this, we will use the set $\textcircled{\Pi}_{k=1}^{3m}(\prod_{i=1}^{3m} a\, b^{n_i}c)$ in place of the simpler set $\textcircled{\Pi}_{i=1}^{3m} ab^{n_i}c$ used in Lemma 3.1. The essence of our argument is contained in

LEMMA 3.3. *Given a sequence of words $T = \langle ab^{n_i}c: 1 \leqslant i \leqslant 3m\rangle$; let*

$$u_i = \prod_{k=1}^{i} ab^{n_k}c \quad \text{and} \quad v_i = \prod_{k=i}^{3m} ab^{n_k}c, \quad \text{for} \quad 1 \leqslant i \leqslant 3m,$$

$$w = u_{3m} = v_1 = \prod_{i=1}^{3m} ab^{n_i}c,$$

$$p = \prod_{i=1}^{3m-1} u_i \quad \text{and} \quad q = \prod_{i=2}^{3m} v_i.$$

Then for any B, $p(a^3b^Bc^3)^m q \in w^{\textcircled{3m}}$ iff $(a^3b^Bc^3)^m \in \textcircled{\Pi}_{i=1}^{3m} ab^{n_i}c$.

Proof. The "if" part of this proof is very simple. We form the p and q of $p(a^3b^Bc^3)^m q$ by collecting the proper prefixes and suffixes of each of the copies of w, in each case leaving one string of the form $ab^{n_i}c$ for a different i to contribute to $(a^3b^Bc^3)^m$. The sequence of middle words obtained in this way will be T, which by hypothesis will shuffle to form the desired word $(a^3b^Bc^3)^m$. To see that the "only if" part holds, we notice that every subword of $p(a^3b^Bc^3)^m q$ the form $a^kb^*c^k$ for $k \in \{1, 3\}$ must have come from exactly k subwords of the form ab^*c, each from a different copy of w. This follows from Lemma 3.1(i). Since $p(a^3b^Bc^3)^m q \in w^{\textcircled{3m}}$, a total of $3m$ copies of each $ab^{n_i}c$ are involved. p and q consume exactly $3m - 1$ copies of each $ab^{n_i}c$ for $1 \leqslant i \leqslant 3m$, leaving exactly one copy of each $ab^{n_i}c$ to be shuffled together to form the middle word $(a^3b^Bc^3)^m$. Thus $(a^3b^Bc^3)^m \in \textcircled{\Pi}_{i=1}^{3m} ab^{n_i}c$. ∎

Using this lemma, we obtain a second *NP* complete problem.

THEOREM 3.2. *Given a finite alphabet* $\Sigma = \{a, b, c\}$ *and words* $w, v \in \Sigma^*$, *the problem of whether or not* $v \in w^{\circledast}$ *is NP complete.*

Proof. Using Lemmas 3.2 and 3.3, given an instance of 3-partition $S = \{n_i : 1 \leqslant i \leqslant 3m\}$ with bound B we can find words w and v such that $v \in w^{(3m)}$ iff S has a 3-partition. Thus we can reduce 3-partition to the above problem. ∎

IV. *NP* Complete Languages and Their Representations

Each of the *NP* complete problems from Section III has the form: given a word and a simple language, is the word contained in the language? In this section we will exhibit fixed languages that contain within them encodings of one of the problems from the previous section. Thus we will convert the problem of membership in which both the word and the language are variable, into the problem of membership in a fixed language. Our basic technique is demonstrated in our first lemma.

LEMMA 4.1. *Given words* v *and* w *over some finite alphabet* Σ *not including the symbols* \$ *and* ¢, *the following are equivalent*:

(i) $w \in v^{\circledast}$,

(ii) $(\$v¢)^k w \in \{\$x¢x : x \in \Sigma^*\}^{\circledast}$ *and* $|w| = k\,|v|$,

(iii) $(\$v¢)^k w^R \in \{\$x¢x^R : x \in \Sigma^*\}^{\circledast}$ *and* $|w| = k\,|v|$ (*where* w^R *indicates the reverse of the word* w),

(iv) $\$v\$^k w \in \bigcup_{x \in \Sigma}(\$x)^{\circledast}$ *and* $|w| = k\,|v|$.

Proof. Obviously (i) ⇒ (ii), we simply shuffle k copies of $\$v¢v$ by collecting the prefixes $\$v¢$ in the front and shuffling the suffixes v to form w. We show that (ii) ⇒ (i) as follows. Since v and w do not contain the letters \$ and ¢, $(\$v¢)^k w \in \{\$x¢x : x \in \Sigma^*\}^{\circledast}$ implies that $(\$v¢)^k w \in ⓟ_{i=1}^{k} \$x_i¢x_i$ for some $x_1, \ldots, x_k \in \Sigma^*$. Thus $(\$v¢)^k \in ⓟ_{i=1}^{k} \$x_i¢x_i'$, where x_i' is a prefix of x_i for i: $1 \leqslant i \leqslant k$. Let $x_i = x_i' x_i''$ for all i and assume that x_i' is not null for some i. Then $\sum_{i=1}^{k} |x_i| + |x_i'| > \sum_{i=1}^{k} |x_i''|$ which implies that $k\,|v| > |w|$. Therefore each x_i' is null. It follows that $x_i = v$ for all i: $1 \leqslant i \leqslant k$, and thus $w \in v^{\circledast}$.

The proof that (i) ⇔ (iii) is very similar to (i) ⇔ (ii). (i) ⇒ (iv) is obvious, since if $w \in v^{\circledast}$, $\$v\$^k w \in (\$v)^{(k+1)}$. To show that (iv) ⇒ (i) we note that if $\$v\$^k w \in (\$x)^{\circledast}$ for some x, then the prefix $\$v$ of $\$v\$^k w$ must be a prefix of $\$x$. Further, $\$v\$^k w$ has $k + 1$ \$'s, hence $\$v\$^k w \in (\$x)^{(k+1)}$. Thus since $|w| = k\,|v|$, v must equal x, and w must arise from shuffling the remaining k copies of v. ∎

THEOREM 4.1. *The languages* $\{\$w¢w : w \in \{a, b, c\}^*\}^{\circledast}$, $\{\$w¢w^R : w \in \{a, b, c\}^*\}^{\circledast}$, *and* $\bigcup_{x \in \{a,b,c\}^*}(\$x)^{\circledast}$ *are NP complete.*

Proof. This follows from Theorem 3.2 and Lemma 4.1. ∎

In the languages of Theorem 4.1, w is a word of a three-letter alphabet. Corollary 4.1 (proven later) shows that two letters suffice.

In Theorem 4.2 we will employ our techniques for converting a variable language membership problem into a fixed language membership problem to the problems considered in Lemma 3.2.

THEOREM 4.2. *The languages* $\{ab^ncde^nf: n \geqslant 0\}^{\circledast}$ *and* $\{a^{n+1}b^nc^nd^n: n \geqslant 0\}^{\circledast}$ *are NP complete.*

Proof. We claim that given natural numbers $B, n_1, \ldots, n_{3n}$ the following conditions hold:

(1) $(d^3e^Bf^3)^m \in \textcircled{\amalg}_{i=1}^{3m} de^{n_i}f \Leftrightarrow$
$w_1 = \prod_{i=1}^{3m} ab^{n_i}c)(d^3e^Bf^3)^m \in \{ab^ncde^nf: n \geqslant 0\}^{\circledast}$.

(2) $(c^Bd^B)^m \in \textcircled{\amalg}_{i=1}^{3m} c^{n_i}d^{n_i} \Leftrightarrow$
$w_2 = (\prod_{i=1}^{3m} a^{n_i+1}b^{n_i})(c^Bd^B)^m \in \{a^{n+1}b^nc^nd^n: n \geqslant 0\}^{\circledast}$.

The forward implications are obvious, we need only select the sets $L_1 = \{ab^{n_i}cde^{n_i}f: 1 \leqslant i \leqslant 3m\}$ and $L_2 = \{a^{n_i+1}b^{n_i}c^{n_i}d^{n_i}: 1 \leqslant i \leqslant 3m\}$ from $\{ab^ncde^nf: n \geqslant 0\}$ and $\{a^{n+1}b^nc^nd^n: n \geqslant 0\}$ and shuffle them together as proscribed in w_1 and w_2, respectively. To prove the reverse implication for (1), assume that $w_1 \in \textcircled{\amalg}_{i=1}^{l} ab^{k_i}cde^{k_i}f$ for some $k_1, \ldots, k_l \in N$. Since w_1 contains $3m$ a's, $l = 3m$. Using Lemma 3.1(i) iteratively, we can show that $\langle k_1, \ldots, k_l \rangle$ is a permutation of $\langle n_1, \ldots, n_{3m} \rangle$ and $(d^3e^Bf^3)^m \in \textcircled{\amalg}_{i=1}^{3m} de^{k_i}f$, which establishes the result. The reverse implication for (2) is proved in a similar manner using Lemma 3.1(iii).

Now using this claim along with Lemma 3.2, we see that we can reduce an instance of 3-partition to a question of membership in the language $\{ab^ncde^nf: n \geqslant 0\}^{\circledast}$ or $\{a^{n+1}b^nc^nd^n: n \geqslant 0\}^{\circledast}$. Hence these languages are *NP* complete. ∎

The language $\{a^{n+1}b^nc^nd^n: n \geqslant 0\}^{\circledast}$ is an interesting borderline case. The extra "a" provides the minimal amount of asymmetry needed to determine the number of words from $\{a^{n+1}b^nc^nd^n: n \geqslant 0\}$ shuffled in forming a word from $\{a^{n+1}b^nc^nd^n: n \geqslant 0\}^{\circledast}$. We could easily replace the extra "a" with a special marker \$, i.e., the language $\{\$a^nb^nc^nd^n: n \geqslant 0\}^{\circledast}$ is also *NP* complete. On the other hand, the corresponding symmetric language $\{a^nb^nc^nd^n: n \geqslant 0\}^{\circledast}$ reduces to $(abcd)^{\circledast}$, which we will show can be recognized in deterministic polynomial time (Theorem 5.1).

Owing to their simple structure, the languages from Theorem 4.2 are of central importance in obtaining results on the power of the operation of iterated shuffle, when used in conjunction with other basic language operations. We will consider the complexity of various classes of languages generated by taking the closure of the finite languages under subsets of the operations shuffle, iterated shuffle, union, concatenation, Kleene star, intersection, intersection with a regular set, non-erasing

homomorphism, and inverse homomorphism, denoted $\odot$, $^{\circledast}$, $+$, $\cdot$, *, $\cap$, $\cap R$, h, h^{-1}, respectively. Thus all our language classes will be contained in *NP*. On the other hand, without iterated shuffle the languages generated are always regular, since the regular languages are closed under all of the remaining operations.

Our next theorem gives numerous sets of operations which can be used to generate *NP* complete languages from finite sets. The examples of *NP* complete languages given will include some of those given above, along with a few variations on these languages, including $\{\$(a\bar{a})^n(b\bar{b})^n(c\bar{c})^n(d\bar{d})^n\colon\ n \geqslant 1\}^{\circledast}$, $\{ab^ncd^nef^ng\colon\ n \geqslant 0\}^{\circledast}$ and $\{\bar{a}ab^n(cd)^ne^nf\bar{f}\colon\ n \geqslant 0\}^{\circledast}$. It can be easily demonstrated that these languages are *NP* complete using the techniques from Theorem 4.2. Before giving the theorem, we prove a few useful technical lemmas.

LEMMA 4.2. *For any* $k \geqslant 1$, $(a_1\bar{a}_1 \cdots a_k\bar{a}_k)^{\circledast} \cap a_1\bar{a}_1 \cdots a_k\bar{a}_k \odot (\bar{a}_1a_1 \cdots \bar{a}_ka_k)^{\circledast} = \{(a_1\bar{a}_1)^n \cdots (a_k\bar{a}_k)^n\colon\ n \geqslant 1\}$.

Proof. This result follows from a simple counting argument involving the ratios of overbarred and nonoverbarred letters in any word in the given intersection. The essence of the argument is given in the proof of Corollary 3.1 part *f* of [6], so we will not repeat it here. ∎

LEMMA 4.3. *Let* $\Sigma = \{a_1, \ldots, a_{2k+1}\}$ *for some* $k > 0$ *and let* $h\colon \Sigma^* \to \bar{\Sigma}^*$ *be the homomorphism defined by* $h(a_1) = \bar{a}_1$, $h(a_{2k+1}) = \bar{a}_{2k}$, $h(a_{2i}) = \bar{a}_{2i}\bar{a}_{2i-1}$ *for* $i\colon 1 \leqslant i \leqslant k$ *and* $h(a_{2i+1}) = \bar{a}_{2i}\bar{a}_{2i+1}$ *for* $i\colon 1 \leqslant i < k$. *Let* $L_1 = h^{-1}((\bar{a}_1 \cdots \bar{a}_{2k})^{\circledast}) \cap a_1(\Sigma - \{a_1\})^*$ *and let* $L_2 = \{a_1a_2^na_3a_4^na_5 \cdots a_{2k-1}a_{2k}^na_{2k+1}\colon\ n \geqslant 0\}$. *Then* $L_1 = L_2$.

Proof. First, assume that we are given $w = a_1a_2^na_3 \cdots a_{2k-1}a_{2k}^na_{2k+1} \in L_2$. $h(w) = \bar{a}_1(\bar{a}_2a_1)^n(\bar{a}_2\bar{a}_3) \cdots (\bar{a}_{2k-2}\bar{a}_{2k-1})(\bar{a}_{2k}\bar{a}_{2k-1})^n\bar{a}_{2k} = (\bar{a}_1\bar{a}_2)^{n+1} \cdots (\bar{a}_{2k-1}\bar{a}_{2k})^{n+1} \in (\bar{a}_1 \cdots \bar{a}_{2k})^{\circledast}$. Hence $w \in h^{-1}((\bar{a}_1 \cdots \bar{a}_{2k})^{\circledast})$. Obviously, $w \in a_1(\Sigma - \{a_1\})^*$, hence $w \in L_1$. Thus $L_2 \subseteq L_1$.

Now, assume that we are given $w = a_1x \in L_1$. Since $h(a_1) = \bar{a}_1$, $h(x) \in \bar{a}_2 \cdots \bar{a}_{2k} \odot \bigcirc\!\!\!\!\!\prod_{i=1}^{n} \bar{a}_1 \cdots \bar{a}_{2k}$ for some $n \geqslant 0$. a_2 and a_3 are the only letters in $\Sigma - \{a_1\}$ whose images under h begin $\bar{a}_1$ or $\bar{a}_2$. Since $h(a_2) = \bar{a}_2\bar{a}_1$ and $h(a_3) = \bar{a}_2\bar{a}_3$, there must exist m: $0 \leqslant m \leqslant n$ such that $x = a_2^ma_3y$, where $h(y) \in \bar{a}_4 \cdots \bar{a}_{2k} \odot \bigcirc\!\!\!\!\!\prod_{i=1}^{m} \bar{a}_3 \cdots \bar{a}_{2k} \odot \bigcirc\!\!\!\!\!\prod_{i=1}^{n-m} \bar{a}_1 \cdots \bar{a}_{2k}$. However, since $y \in (\Sigma - \{a_1\})^*$, $\bar{a}_1$ occurs in $h(y)$ only if a_2 occurs in y, in which case $\bar{a}_1$ occurs immediately proceeded by $\bar{a}_2$, because $h(a_2) = \bar{a}_2\bar{a}_1$. But every occurrence of $\bar{a}_2$ in $h(y)$ is proceeded by some occurrence of $\bar{a}_1$, and thus no $\bar{a}_1$ can occur in $h(y)$. This implies that $n - m = 0$, i.e., $x = a_2^na_3y$ and $h(y) \in \bar{a}_4 \cdots \bar{a}_{2k} \odot \bigcirc\!\!\!\!\!\prod_{i=1}^{n} \bar{a}_3 \cdots \bar{a}_{2k}$. In this case $y \in (\Sigma - \{a_1, a_2, a_3\})^*$ and a_4 and a_5 are the only letters in $\Sigma - \{a_1, a_2, a_3\}$ whose images under h begin with $\bar{a}_3$ or $\bar{a}_4$. Hence we can repeat the above argument. Continuing in this manner, it is apparent that $x = a_2^na_3 \cdots a_{2k}^na_{2k+1}$, and thus $w \in L_2$. Hence $L_1 = L_2$. ∎

LEMMA 4.4. *Let* $\Delta = \Sigma \cup \{\$\}$, *where* $\$ \notin \Sigma$. *Then for any* $L \subseteq \Delta^*$, *if* $(L \cap \$\Sigma^*)^{\circledast}$

is NP complete, then there exists an alphabet Γ and a non-erasing homomorphism $h: \Delta^* \to \Gamma^*$ such that $(h(L \cap (\Delta^* - \Sigma^+)))^{\circledast}$ is NP complete.

Proof. Let $\Gamma = \Delta \cup \{\mathbb{Q}\}$, where $\mathbb{Q} \notin \Delta$. Let h be the homomorphism defined by $h(\$) = \mathbb{Q}\$$ and $h(a) = a$ for $a \in \Sigma$. Let $T = (h(L \cap (\Delta^* - \Sigma^+)))^{\circledast} \cap \mathbb{Q}^*\Delta^*$. Since all nonempty words in $\Delta^* - \Sigma^+$ contain an occurence of \$ and h is non-erasing, $T = \bigcup_{m=0}^{\infty} \mathbb{Q}^m (L \cap \$\Sigma^*)^{\circledast}$.

Now assume that there is a deterministic polynomial time algorithm for T. Since $w \in (L \cap \$\Sigma^*)^{\circledast}$ if and only if $\mathbb{Q}^m w \in T$ for some $m: 0 \leqslant m \leqslant |w|$, we can easily convert this algorithm into a deterministic polynomial time algorithm for $(L \cap \$\Sigma^*)^{\circledast}$. Since this language is *NP* complete, it follows that T is *NP* hard [3]. Since T is obviously in the class *NP*, it follows that T is *NP* complete. Finally, since T is *NP* complete, it follows that $(h(L \cap (\Delta^* - \Sigma^+)))^{\circledast}$ is *NP* complete. ∎

THEOREM 4.3. *There exists a finite alphabet Σ such that the closure of the class of finite languages in Σ^* under each of the following sets of operations contains NP complete languages*:

(1) $\{\circledast, \cap R\}$,

(2) *any set in* $\{\circledast\} \times \{\odot, \cdot\} \times \{\cap, h, h^{-1}\}$ *or* $\{\circledast\} \times \{\cap\} \times \{h, h^{-1}\}$ *and* $\{\circledast, h, h^{-1}\}$.

Proof. (1) This follows from the fact that $(\{\$, abcd\}^{\circledast} \cap \$a^*b^*c^*d^*)^{\circledast} = \{\$a^n b^n c^n d^n : n \geqslant 0\}^{\circledast}$. (See Theorem 4.2 and comments following.)

(2) For each of the nine sets of operations listed we give an example of an *NP* complete language generated from finite sets using these operations. Detailed verification of the examples is left to the reader.

(i) $\{\circledast, \cdot, \cap\}$.

$(a \cdot (abcd)^{\circledast} \cap a^{\circledast} \cdot b^{\circledast} \cdot c^{\circledast} \cdot d^{\circledast})^{\circledast} = \{a^{n+1} b^n c^n d^n : n \geqslant 0\}^{\circledast}$.

(ii) $\{\circledast, \odot, \cap\}$.

$((\$ \odot (a\bar{a}b\bar{b}c\bar{c}d\bar{d})^{\circledast}) \cap (\$a\bar{a}b\bar{b}c\bar{c}d\bar{d} \odot (\bar{a}a\bar{b}b\bar{c}c\bar{d}d)^{\circledast}))^{\circledast} = \{\$(a\bar{a})^n (b\bar{b})^n (c\bar{c})^n (d\bar{d})^n : n \geqslant 1\}^{\circledast}$. This follows using Lemma 4.2.

(iii) $\{\circledast\} \times \{\odot, \cdot\} \times \{h^{-1}\}$.

Let $\Sigma = \{a, b, c, d, e, f, g\}$ and let $k: \Sigma^* \to \{\bar{\Sigma} \cup \{\$\}\}^*$ be the homomorphism defined by $k(a) = \$\bar{a}$, $k(b) = \bar{b}\bar{a}$, $k(c) = \bar{b}\bar{c}$, $k(d) = \bar{d}\bar{c}$, $k(e) = \bar{d}\bar{e}$, $k(f) = \bar{f}\bar{e}$ and $k(g) = \bar{f}$. Let $\square$ be an operator in $\{\odot, \cdot\}$ and let $L = \$\square(\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}$. Since every word in L contains exactly one occurrence of \$, any word in $k^{-1}(L)$ must contain exactly one occurrence of a. Since every word in L begins with either \$ or $\bar{a}$ and no word in the range of k begins with $\bar{a}$, every word in $k^{-1}(L)$ must begin with a. Thus $k^{-1}(L) = k^{-1}(L) \cap a(\Sigma - a)^*$. Let h be a homomorphism which is identical to k except that $h(a) = \bar{a}$. Since h and k are identical on $\Sigma - \{a\}$, it is easily verified that $k^{-1}(L) \cap a(\Sigma - a)^* = h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap a(\Sigma - a)^*$. Hence $k^{-1}(L) = h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap a(\Sigma - a)^*$. Since h is identical to the homomorphism given in Lemma 4.3 with $a, b, c, \ldots$, corresponding to

$a_1, a_2, a_3 \ldots$, respectively, it follows that $k^{-1}(L) = \{ab^n cd^n ef^n g: n \geqslant 0\}$. Hence $(k^{-1}(L))^{\circledast}$ is the *NP* complete language $\{ab^n cd^n ef^n g: n \geqslant 0\}^{\circledast}$.

(iv) $\{\circledast, h^{-1}, \cap\}$.

Let Σ and h be defined as in (iii). Let $j: \Sigma^* \to \bar{\Sigma}^*$ be the homomorphism defined by $l(a) = \bar{a}$, $l(x) = \lambda$ for $x \in \Sigma - \{a\}$. Then $(h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap l^{-1}(\bar{a}))^{\circledast} = (h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap a(\Sigma - \{a\})^*)^{\circledast} = \{ab^n cd^n ef^n g: n \geqslant 0\}^{\circledast}$, again using Lemma 4.3.

(v) $\{\circledast, h, h^{-1}\}$.

Again let Σ and h be defined as in (iii). Using Lemma 4.3, it follows that $(h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap a(\Sigma - \{a\})^*)^{\circledast} = \{ab^n cd^n ef^n b: n \geqslant 0\}^{\circledast}$ which is *NP* complete. Hence by Lemma 4.4 there exists a non-erasing homomorphism l such that $(l(h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \cap (\Sigma^* - (\Sigma - \{a\})^+)))^{\circledast}$ is *NP* complete. However, $h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast}) \subseteq (\Sigma^* - (\Sigma - \{a\})^+)$, hence $(l(h^{-1}((\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f})^{\circledast})))^{\circledast}$ is *NP* complete.

(vi) $\{\circledast, h, \cap\}$. Let $\Sigma = \{a, b, c, d\}$ and $\Delta = \Sigma \cup \{\$\}$.

Let $L = \{\$, a\bar{a}b\bar{b}c\bar{c}d\bar{d}\}^{\circledast} \cap \{\$a\bar{a}b\bar{b}c\bar{c}d\bar{d}, \bar{a}a\bar{b}b\bar{c}c\bar{d}d\}^{\circledast}$. Using Lemma 4.2, it is easily verified that $(L \cap \$\Sigma^*)^{\circledast} = \{\$(a\bar{a})^n(b\bar{b})^n(c\bar{c})^n(d\bar{d})^n: n \geqslant 1\}^{\circledast}$, which is *NP* complete. Hence by Lemma 4.4 there exists a non-erasing homomorphism h such that $(h(L \cap (\Delta^* - \Sigma^+)))^{\circledast}$ is *NP* complete. However $L \subseteq \Delta^* - \Sigma^+$, hence $(h(L))^{\circledast}$ is *NP* complete.

(vii) $\{\circledast\} \times \{\odot, \cdot\} \times \{h\}$.

Let $\Sigma = \{a, b, c, d, e, f\}$ and let $h: \Sigma^* \to (\Sigma \cup \bar{\Sigma})^+$ be the homomorphism defined by $h(a) = \bar{a}a$, $h(b) = b$, $h(c) = cd$, $h(d) = d$, $h(e) = e$, $h(f) = f\bar{f}$. Let $\square$ be an operation in $\{\odot, \cdot\}$. Let $L = (h(a \square (bce)^{\circledast} \square f))^{\circledast}$. To prove that L is *NP* complete we intersect it with the regular set $R = \bar{a}^*(ab^*c)^*(dc)^*(d^3e^*f^3)^*\bar{f}^*$ and observe that $L \cap R = L' \cap R$, where $L' = \{\bar{a}ab^n(cd)^n e^n f\bar{f}: n \geqslant 0\}^{\circledast}$. It suffices to show that $L' \cap R$ is *NP* complete. To do this we use the following equivalence combines with Theorem 3.1:

$$\bar{a}^{3m}\left(\prod_{i=1}^{3m} ab^{n_i}c\right)(dc)^{mB-3m}(d^3e^Bf^3)^m\bar{f}^{3m} \in L' \quad \text{iff} \quad (d^3e^Bf^3)^m \in \textcircled{\scriptsize \amalg}_{i=1}^{m} de^{n_i}f.$$

This completes the proof of Theorem 4.3. ∎

The results of Theorem 4.3 are summarized in Table I. For each pair of operations in the set $\cdot, \cap R, \odot, \cap, h, h^{-1}$ an indication of the complexity of the languages generated using these operations in conjunction with iterated shuffle is given. The entry in the row and column for operations O_1 and O_2 indicates the complexity of the closure of the finite sets under the operations $\circledast$, O_1, and O_2. The entry "c" denotes the fact that the corresponding language class contains *NP* complete languages, while the entry "$\mathscr{P}$" denotes the fact that all languages of this class can be recognized in deterministic polynomial time. (The latter results are obtained in Corollary 5.1 in Section V.) The entry "?" denotes the fact that it could not be shown that the corresponding language class contains *NP* complete languages nor that it is contained within $\mathscr{P}$. We leave it to the interested reader to extend the results of Theorem 4.3 to

TABLE I

Operations	$\cdot$	$\cap R$	$\odot$	$\cap$	h	h^{-1}
$\cdot$	?[a]	$\mathscr{C}$	?	$\mathscr{C}$	$\mathscr{C}$	$\mathscr{C}$
$\cap R$		$\mathscr{C}$	$\mathscr{C}$	$\mathscr{C}$	$\mathscr{C}$	$\mathscr{C}$
$\odot$			$\mathscr{P}$	$\mathscr{C}$	$\mathscr{C}$	$\mathscr{C}$
$\cap$				$\mathscr{P}$	$\mathscr{C}$	$\mathscr{C}$
h					$\mathscr{P}$	$\mathscr{C}$
h^{-1}						$\mathscr{P}$

[a] Jantzen [7] recently proved this class to be in $\mathscr{P}$.

include language classes that can be described using the operations union, Kleene star, or any other interesting operation that does not involve cancellation.

In most of our *NP* completeness results up to this point, we have relied on alphabets of size three or larger. It is obvious that with the alphabet $\Sigma = \{a\}$, Theorems 3.1 and 3.2 do not hold, and that all languages generated from a^* using the operations considered above, excluding h and h^{-1}, are regular. The case in which Σ has just two elements remains unexplored. The possibility remains that there is some sort of complexity gap between alphabets of size two and three in the context of some of our iterated shuffle results. The following indicates that this is not the case.

LEMMA 4.5. *Given the alphabet* $\Delta = \{a_1, \ldots, a_k\}$ *we define the homomorphism* $h: \Delta^* \to \{ab\}^*$ *as* $h(a_i) = a^{i+1}b^i$. *Then for any languages* $L_1, L_2 \subset \Delta^*$ *we have*

(i) $w \in L_1^{\circledast}$ *iff* $h(w) \in (h(L_1))^{\circledast}$,

(ii) $w \in L_1 \odot L_2$ *iff* $h(w) \in h(L_1) \odot h(L_2)$,

(iii) $w \in L_1 \cdot L_2$ *iff* $h(w) \in h(L_1) \cdot h(L_2)$,

(iv) $w \in L_1 + L_2$ *iff* $h(w) \in h(L_1) + h(L_2)$,

(v) $w \in L_1 \cap L_2$ *iff* $h(w) \in h(L_1) \cap h(L_2)$.

Proof. (iii)–(v) follow from the fact that h is a code, i.e., the function $h: \Delta^* \to \{a, b\}^*$ is injective. For parts (i) and (ii), it is obvious that if $w \in L_1^{\circledast}$ then $h(w) \in (h(L_1))^{\circledast}$ and if $w \in L_1 \odot L_2$ then $h(w) \in h(L_1) \odot h(L_2)$, we need only shuffle images of letters in Δ as units. For the reverse implications we use Lemma 3.1(iii) iteratively to "decode" a word in the range of h formed by shuffling smaller words from the range of h. ∎

The "shuffle resistant" code defined in the above theorem can be used to reduce the size of the alphabet required for our results.

COROLLARY 4.1. *Theorems* 3.1, 3.2, 4.1, 4.2 *and the parts of* 4.3 *not involving the operations* h *and* h^{-1} *hold for* $\Sigma = \{a, b\}$.

Proof. This follows directly from Lemma 4.5. ∎

For example, the *NP* complete language $\{ab^n cde^n f: n \geqslant 0\}^{\circledast}$ can be transformed into the *NP* complete language $\{a^2b(a^3b^2)^n a^4b^3a^5b^4(a^6b^5)^n a^7b^6: n \geqslant 0\}^{\circledast}$, using Lemma 4.5. ∎

V. Language Classes Contained in *P*

In Section IV, we have presented a variety of *NP* complete languages, each of which had the form $L^{\circledast}$ for some language L. In each case there is a strong structural relationship between the language L and the language $\{ww: w \in \Sigma^*\}$. Each of the languages exhibits some kind of unbounded pairing or repetition along with markers, which in the simplest case reduces to counting as in $\{ab^n cde^n f: n \geqslant 0\}$. Hence there are *NP* complete languages among the iterated shuffles of linear languages. In Theorem 5.1 we consider the complexity of languages obtained from the iterated shuffle of regular languages.

THEOREM 5.1. *For any regular R, the language $R^{\circledast}$ can be recognized in deterministic polynomial time.*

Proof. Let us suppose we are given a finte automaton A for the language R with k states, possibly nondeterministic, but without λ-moves. To recognize a word $w \in R^{\circledast}$, we imagine that we begin with a pile of pebbles placed in the start state of A. Each time we read a letter l from w, we must find a state q of A which has one or more pebbles in it, such that q maps with letter l to some state q' in the automaton A. Upon finding such a state q, we remove a pebble from q and place it in q'. We claim that $w \in R^{\circledast}$ iff there is a way to move some initial pile of pebbles while reading w such that when we are finished, all the pebbles are in a final state. This follows since the movement of each pebble contributes one word from R, which is shuffled with the words obtained from moving the other pebbles.

We will determine if $w \in R^{\circledast}$, for $w \neq \lambda$, by computing all possible final pebble configurations for w. If $|w| = n$ then we need to start with at most n pebbles in the start state. At any given time as we are reading a letter of w, there are at most n pebbles in any given state. Since A has k states, the number of possible intermediate pebblings of the automaton A is less than n^k. Hence, we can keep track of all possible pebble configurations of A as we read w. Each time we read a letter from w we revise our list of possible configurations using $O(n^k)$ time. Thus our total time is $O(n \cdot n^k) = O(n^{k+1})$, which completes the proof. ∎

COROLLARY 5.1. *For any finite alphabet, the closure of the finite languages in Σ^* under any set of operations in $\{\circledast\} \times \{+, *, \cap, h, h^{-1}\}$ is contained in $\mathscr{P}$, the class of all languages recognized in deterministic polynomial time.*

Proof. The following calculation rules are easily verified. For any languages L, T, $L_1, \ldots, L_k \subseteq \Sigma^*$ and homomorphisms $h_1: \Sigma^* \to \Delta^*$ and $h_2: \Delta^* \to \Sigma^*$,

(i) $(L^{\circledast} + T^{\circledast})^{\circledast} = (L + T^{\circledast})^{\circledast} = (L + T)^{\circledast}$,

(ii) $(L^*)^{\circledast} = (L^{\circledast})^* = (L^{\circledast})^{\circledast} = L^{\circledast}$,

(iii) $(\bigcap_{i=1}^{k} L_i^{\circledast})^{\circledast} = \bigcap_{i=1}^{k} L_i^{\circledast}$,

(iv) $(h_2^{-1}(L^{\circledast}))^{\circledast} = h_2^{-1}(L^{\circledast})$, and

(v) $(h_1(L^{\circledast}))^{\circledast} = (h_1(L))^{\circledast}$.

Using these rules, we can easily find a representation for any language in any of the above classes in which the applications of the operation $^{\circledast}$ do not appear nested. Hence the result follows from Theorem 5.1, using the elementary closure properties of the class $\mathscr{P}$. ∎

Following Jantzen [6], we make the following definitions:

DEFINITION. Given a finite alphabet Σ, we define the language class Shuf_Σ as the closure of the finite languages over Σ under the operations $+$, $\odot$, and $\circledast$ and the language class SE_Σ as the closure of the finite languages over Σ under the operations $+$, $\cdot$, *, $\odot$, and $\circledast$.

The class SE_Σ is the class of languages definable by the shuffle expressions of [12] or the c-expressions of [8] over the alphabet Σ. In [6] Jantzen shows that $\text{Shuf}_\Sigma \not\subseteq SE_\Sigma$, $|\Sigma| \geqslant 2$, and he gives a representation for an arbitrary $L \in \text{Shuf}_\Sigma$ as $\bigcup_{i=1}^{m} M_i \odot N_i^{\circledast}$ for finite $M_i, N_i \subset \Sigma^*$. A language of the form $R \odot S^{\circledast}$ for regular sets R and S can be recognized in deterministic polynomial time by a simple extension of the algorithm of the previous theorem. Thus we have

COROLLARY 5.2. *For any finite alphabet Σ, the class* Shuf_Σ *is contained in* $\mathscr{P}$.

The complexity of the class SE_Σ for $|\Sigma| \geqslant 2$ remains open, as does the complexity of any of the classes of languages generated by closure of the finite languages under any subset of the operations $+$, $\cdot$, *, $\odot$, and $\circledast$ which includes both $\cdot$ and $\circledast$. We have not found any *NP* complete languages in the class SE_Σ for any Σ, nor have we been able to exhibit deterministic polynomial time algorithms for any class of languages containing the finite languages, and closed under $\cdot$ and $\circledast$. Our hypothesis is that such polynomial time algorithms can be found, and indeed that SE_Σ is contained in $\mathscr{P}$ for any[2] Σ.

ACKNOWLEDGMENTS

We would like to thank Andrzej Ehrenfeucht for numerous helpful discussions concerning this material and Matthias Jantzen for suggesting several improvements on an earlier draft of this paper.

[2] Jantzen [7] showed recently that the closure of the finite languages under the operations $+$, $\cdot$, *, and $\circledast$ generates languages which are contained in $\mathscr{P}$.

REFERENCES

1. T. ARAKI, T. KAGIMASA, AND N. TOKURA, Relations of flow languages to Petri net languages, *Theoret. Comput. Sci.* **15** (1) (1981), 51–76.
2. M. GAREY AND D. JOHNSON, Complexity results for multiprocessor scheduling under resource constraints, *SIAM J. Comput.* **4** (1975), 397–411.
3. M. GAREY AND D. JOHNSON, "Computers and Intractability. A Guide to the Theory of *NP* Completeness," Problem SP 15, p. 224, Freeman, San Francisco, 1980.
4. J. GISHER, Shuffle languages, Petri nets, and context-sensitive grammars, *Comm. ACM* **24** (9) (1981), 597–605.
5. M. JANTZEN, "Eigenschaften von Petrinetzsprachen,' Doctoral Disseration, Bericht Nr. IFI-HH-B-64, Fachbereich Informatik, Universitat Hamburg, 1979.
6. M. JANTZEN, The power of synchronizing operations on strings, *Theoret. Comput. Sci.* **14** (2) (1981), 127–154.
7. M. JANTZEN, Extending regular expressions with iterated shuffle, unpublished manuscript, Fachbereich Informatik, Universitat Hamburg, 1982.
8. T. KIMURA, An algebraic system for process structuring and interprocess communication, *in* "Proceedings 8th Annual ACM Symp. on Theory of Computing," pp. 92–100, 1976.
9. W. F. OGDEN, W. E. RIDDLE, AND W. C. ROUNDS, Complexity of expressions allowing concurrency, *in* "Proceedings, 5th ACM POPM," Tucson, Arizona, pp. 185–194, 1978.
10. W. E. RIDDLE, "Modelling and Analysis of Supervisor Systems," Ph. D. thesis, Computer Science Dept., Stanford University, 1972.
11. A. C. SHAW, System design and documentation using path expressions, *in* "Proceedings, Sagamore Computer Conference on Parallel Processing," pp. 180–181, IEEE Computing Society, 1975.
12. A. C. SHAW, Software description with flow expression, *IEEE Trans. Software Engrg.* 3, **SE-4** (1978), 242–254.
13. G. SLUTZKI, Descriptional complexity of concurrent processes, unpublished manuscript, Department of Mathematics and Computer Science, Clarkson College of Technology, Potsdam, New York, 1980.